REMARKS

Claims 1-30, 37-39 are pending in this application, claims 31-36 having been previously canceled. Applicant amends claims 1, 3, 5-9, 14-16, 18-23, and 37-39 herein. Applicant cancels claims 2, 4, and 17 without prejudice or disclaimer. Applicant adds new claim 40. Applicant urges that all of the pending claims are in condition for allowance. Applicant respectfully requests reconsideration of the outstanding rejections and allowance of all pending claims in view of the reasons set forth below.

The Office Action of January 16, 2008 included a final rejection of claims 1-30 and 37-39. A Response was filed on March 17, 2008, which was followed by an Advisory Action on May 21, 2008, reiterating the rejection of the January 16 Office Action. Therefore, Applicant submits a Request for Continued Examination in conjunction with this Amendment.

Applicant amends the claims to consistently use the word "said" as opposed to "the." Applicant further amends the claims to remove the phrase "the step of" and "the further step of." Applicant amends the claims to recite a "computer implemented method" rather than a "method" and a "computer readable storage medium for storing computer executable instructions" rather than a "computer readable medium" for storing "instructions." Further, Applicant amends the claims to use the words "a" and "the" where necessary for antecedent purposes.

Applicant amends the independent claims to recite "calling said assignment function from said language processor" for clarity.

I. Claim Rejection under 35 U.S.C. § 102(b)

In the Office Action of January 16, 2008, claims 1-11, 13, 16-26, 28, and 39 were rejected under 35 U.S.C. § 102(b) as being anticipated by Conway ("Parsing with C++ deferred expressions," ACM SIGPLAN Notices, vol. 29, no. 9, pp. 9-16, ACM, 1994), hereafter "Conway" (Office Action at page 2). Applicant respectfully traverses the rejection.

A. Claim 1

Amended claim 1 recites:

1. In a program development environment, a computer implemented method comprising:

providing, via a programming language, a language processor with built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including methods that retrieve values for base language objects;

defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;

defining said assignment function in more than one class;

overloading said assignment function based on a context of said base language objects; and

calling said assignment function from said language processor to determine a value of at least one assignment within at least one of said base language and a base language extension to said base language.

Amended claim 1 recites the feature of ***overloading said assignment function based on a context of said base language objects***, a feature formerly found in dependent claim 4. Applicant respectfully submits that Conway fails to disclose this feature of claim 1.

The Examiner suggests (in relation to former claim 4) that Conway discloses this feature at page 12, §5 ("Embedding Deferred Assignments") last paragraph, and at page 10, §4 ("Embedding References to Grammar Components"), second paragraph (Office Action at page 3). As §4 of Conway begins on page 11, Applicants assume that the Examiner means Conway at page 11, §4, second paragraph.

Conway's assignment function is not overloaded ***based on a <u>context of said base language objects</u>***, as recited in claim 1. There is no discussion in §§4-5, nor any code in Figures 3-5 (cited by the Examiner), to suggest that Conway takes the context of a base language object into account. Rather, Conway discloses an opposite approach; in Conway, the assignment operator is overloaded without reference to the context of the base language object. Conway states at §5:

> The *ParserAssignment* class is derived from *ParserExpr* and stores pointers to a *ParserVar* (the target of the assignment) and a *Parser Expr* (representing the value to be assigned) in its *lhs* and *rhs* members. When a *ParserAssignment* object is evaluated, it executes the deferred assignment, as indicated in Figure 4.

> As is standard in C and C++, the evaluation of the assignment returns the assigned value, so that deferred assignments may be chained or otherwise embedded in larger expressions.

This passage describes that the evaluation of the assignment only returns the assigned value. The context of the base language objects (the objects pointed to by the *lhs* and *rhs* pointers, in this case) is not taken into account in the assignment function. The assignment is simply executed.

Conway continues:

> Similar classes representing other deferred assignment operations can easily be created by adding the corresponding overloaded operator to the *ParserVar* class, creating the corresponding deferred assignment class, and redefining the *Evaluate()* method of the new class to perform the appropriate operation.

This passage describes overloaded operators, but does not discuss overloading the assignment operator ***based on a context of said base language objects***. Context is not mentioned in relation to the overloaded operator.

Conway does reference an object called "Context c" in the assignment operator (Conway at Figure 2). However, "Context c" is not ***a <u>context of said base language objects</u>***, which is present in claim 1. Rather, Context c is a context <u>of the grammar</u> (Conway at §4, "Embedding References <u>to Grammar Components</u>"). Conway states at §4 that "when the Evaluate() method … is called, it interrogates the Context object passed as its argument and returns the value of <u>the appropriate component of the parent [grammar] rule</u>." "Context c" is a grammar rule, and not a context of the base language objects. Thus, even if Conway's assignment function were overloaded based on "Context c," Conway would still not disclose ***overloading said assignment function based on <u>a context of said base language objects</u>***.

Therefore, Conway does not disclose each and every element of independent claim 1. Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §102(b) rejection of claim 1.

B. <u>Claims 2-11 and 13</u>

Claims 3, 5-11 and 13 depend from claim 1 and thus include all of the elements of claim 1. Therefore, Applicant submits that claims 3, 5-11 and 13 are also in condition for allowance, and requests that the Examiner withdraw the 35 U.S.C. §102(b) rejection of claims 2-11 and 13. Because claims 2 and 4 have been canceled, Applicant considers the rejection of those claims to be moot.

Further, claim 5 recites ***evaluating said class at compile-time, <u>and adjusting said resulting class definitions from said evaluation to increase efficiency of run-time performance</u>***. There is no disclosure of ***adjusting said resulting class definitions from said evaluation to increase efficiency of run-time performance*** in Conway. No adjustment to the class definitions is made from the evaluation in Conway. Further, Conway is silent concerning the efficiency of run-time performance. Conway is not concerned with the classes that the grammar will be used in conjunction with, but rather the grammar itself. Therefore, Conway does not disclose adjusting class definitions at compile time.

For at least the reasons described above, Applicant submits that claim 5 is in condition for allowance, and requests that the Examiner withdraw the 35 U.S.C. §102(b) rejection of claim 5.

C. <u>Claims 16-26, 28 and 39</u>

Independent claim 16 recites ***one or more instructions for overloading said assignment function <u>based on a context of said base language objects</u>***, and independent claim 39 recites ***overloading a mathematical operator with said assignment function <u>based on a context of a plurality of said base language objects</u>***. Claims 16 and 39 are therefore allowable for the same reasons set forth above with regard to claim 1. Therefore, Applicant requests that the Examiner withdraw the 35 U.S.C. §102(b) rejection of claims 16 and 39.

Because claim 17 has been canceled, Applicant considers the rejection of this claim to be moot.

Claims 18-26 and 28 depend from claim 16 and thus include all of the elements of claim 16. Further, claim 20 recites the same elements as claim 5, which further define patentable differences over the cited prior art, as discussed above.

Therefore, Applicant submits that claims 18-26 and 28 are also in condition for allowance and requests that the Examiner withdraw the 35 U.S.C. §102(b) rejection of claims 18-26 and 28.

II. Claim Rejection under 35 U.S.C. § 103(a)

In the Office Action, claims 12, 14, 15, 27, 29, 30 and 37-38 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Conway (Office Action at page 7). Applicant respectfully traverses the rejection.

The Examiner suggests that, while Conway did not disclose the limitations of claim 12, and claims 29 and 30, the limitations would have been obvious modifications to the Conway system (Office Action at page 7). The Examiner rejects claims 14 and 15 as corresponding to claims 29 and 30, respectively. The Examiner also rejects claim 27 as corresponding to claim 12 and rejects claims 37 and 38 for the same reasons as he rejected claims 14 and 15.

Claims 14-15 and 37-38 are independent claims and recite ***overloading said assignment function <u>based on a context of said base language objects</u>***. As discussed above in connection with claim 1, this claim feature is not disclosed by Conway. This feature is further not suggested by Conway, which is directed to the design of class parser libraries, and is not concerned with the classes that they operate on. Accordingly, Conway does not support a valid 35 U.S.C. §103 rejection of claims 14-15 and 37-38. Applicant requests that the Examiner withdraw the 35 U.S.C. §103(a) rejection of claims 14-15 and 37-38.

Claims 12, 27 and 29-30 are dependent claims which depend from independent claims reciting ***overloading said assignment function <u>based on a context of said base language objects</u>***. Claims 12, 27 and 29-30 are therefore also in condition for allowance for the same reasons set forth above with respect to claims 14-15 and 37-38. Applicant therefore respectfully requests that the Examiner withdraw the 35 U.S.C. §103(a) rejection of claims 12, 27 and 29-30.

III. <u>New Claim 40</u>

Applicant adds new claim 40, which recites ***evaluating said second parse tree object to obtain a value, said evaluating done <u>based on a context provided by said first parse tree object</u>***. Applicant respectfully submits that the cited prior art does not teach or suggest at least this element of new claim 40. Thus, Applicant respectfully submits that new claim 40 is in condition for allowance.

CONCLUSION

In view of the above Remarks, Applicant believes the pending application is in condition for allowance and urges the Examiner to pass all of the pending claims to allowance. Should the Examiner feel that a teleconference would expedite the prosecution of this application, the Examiner is urged to contact the Applicant's attorney at (617) 227-7400.

Please charge any shortage or credit any overpayment of fees to our Deposit Account No. 12-0080, under Order No. MWS-039RCE. In the event that a petition for an extension of time is required to be submitted herewith, and the requisite petition does not accompany this response, the undersigned hereby petitions under 37 C.F.R. §1.136(a) for an extension of time for as many months as are required to render this submission timely. Any fee due is authorized to be charged to the aforementioned Deposit Account.

Dated: July 16, 2008

Respectfully submitted,

Electronic signature: /Kevin J. Canning/
Kevin J. Canning
Registration No.: 35,470
LAHIVE & COCKFIELD, LLP
One Post Office Square
Boston, Massachusetts 02109-2127
(617) 227-7400
(617) 742-4214 (Fax)
Attorney/Agent For Applicant